GSoC 2015 Ashan Dhananjaya(ashandk) Proposal
Apache STRATOS

Description

Ashan Dhananjaya - Showing health statistics in GUI (STRATOS-1244)

Apache Stratos is a highly-extensible Platform-as-a-Service (PaaS) which include multitenancy, Multi fractured auto scaling, Cloud bursting and Scalable dynamic load balancing. It consumes a lot of processing power and memory but this product currently doesn't have a UI to show health statistics of its product (clusters) and instances.This project implements a user friendly UI which enable developers to easily access, understand, and use data through visualization.

Deliverables

Priorities are divided among
Optional - may not be implemented, but can be a implemented in future.
High Priority - will most probably be implemented.
Critical - Will be implemented.

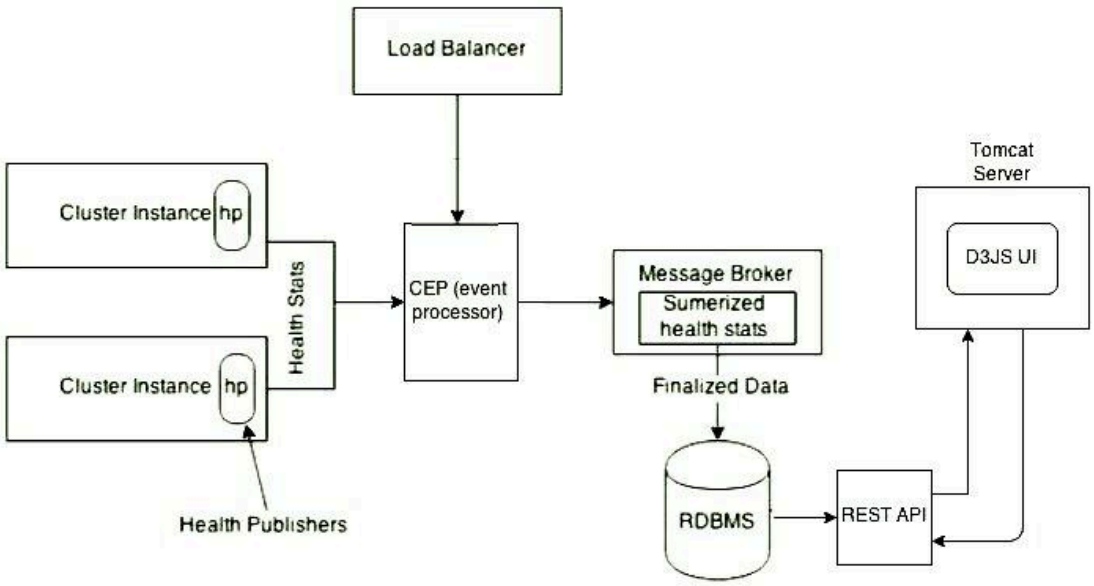
- ❏ [Critical] A REST API to expose data and which can invoke/query data.
- ❏ [Critical] Load balancer Statistics (Requests InFlight) per each an every individual Cluster (Product).
- ❏ [Critical] Statistics of each and every individual member.
- ❏ [High Priority] Time Line of each and every individual Cluster which contains the total statistic data of its member.
- ❏ [High Priority] Logs of the Clusters.

The Big Picture

A Health Publisher is included in each instance spawned in the Cartridge Agent component.it publishes health statistics of each cluster instance such as load average of an instance, memory consumption, Load Average Event etc. These summarized health stats will be exposed using the REST API and the API will enable to query the data.

Apache Stratos uses WSO2 Complex Event Processor (CEP) to process it's real time data and it will be published through a thrift protocol. Load Balancer also publishers statistics

(e.g., the failed message count for a particular cluster, serving request count for that cluster and more) to CEP.



CEP process all the data in real-time and publishes the summarized health statistics events, which include the derivative, gradients and the average of each metric for each instance, to the Summarized Health Stats topic in the Message Broker.By creating a RDBMS datasourcethe Message Broker can write all the sumarized data into the database.Using the REST APIit can retrieve data as json objects from the RDBMS. Using D3JS it will be easy to visualize the graphs of each and every stat.

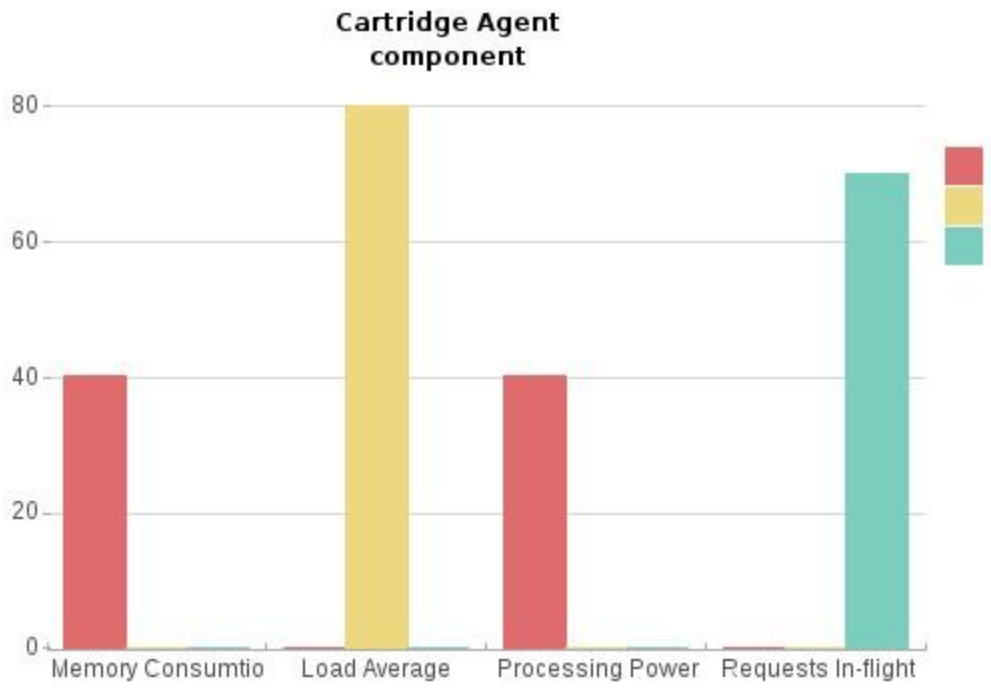
The Sketch for the UI

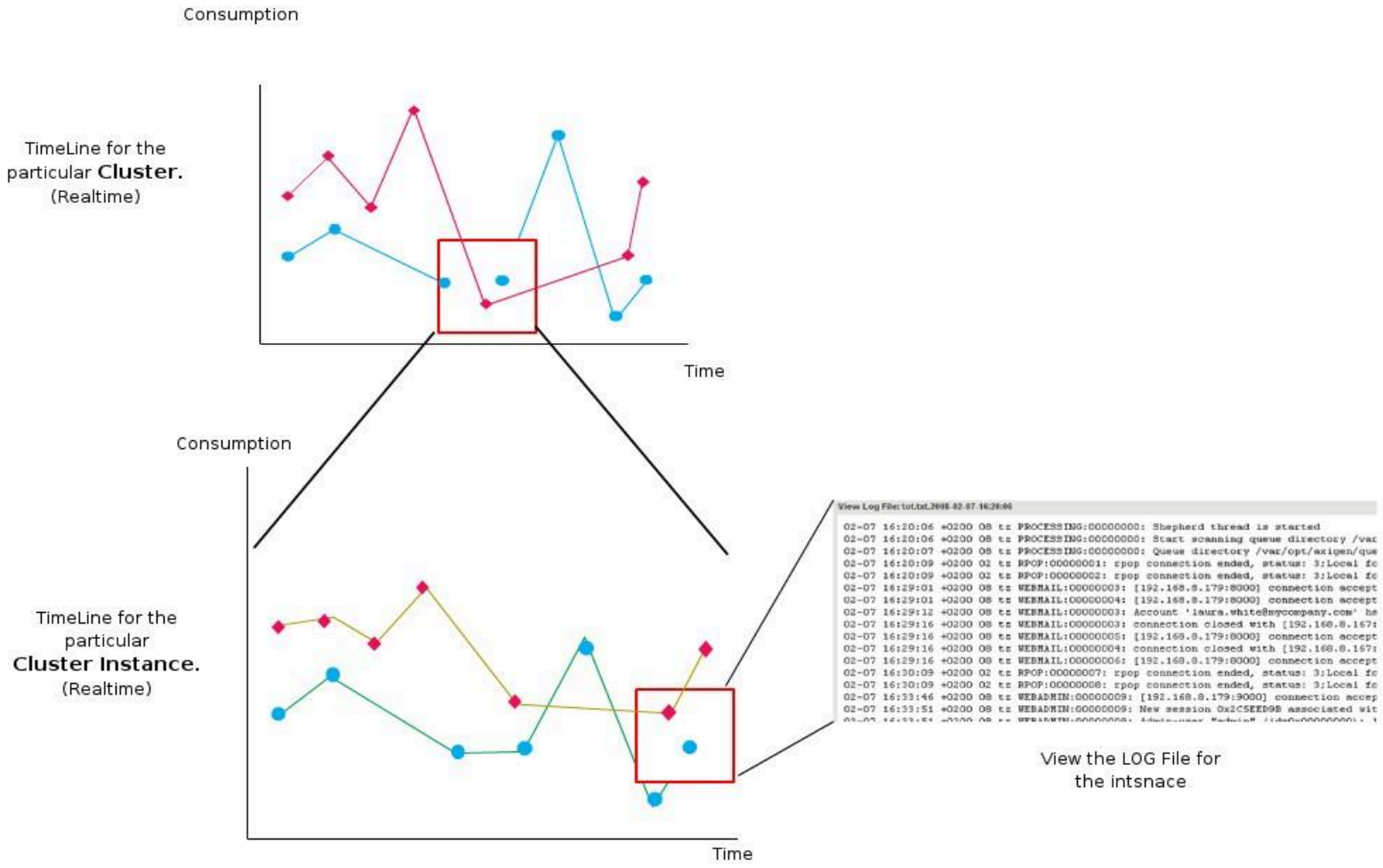
Sketch [1]

The main view of the web application will show the overall usage of the Health Statistics. The timeline will show the logs of the particular time of the graph. And there will be tabs to direct directly to Cartridge Clusters or Cartridge cluster instances.

Also by clicking on the Cartridge cluster instance it will automatically show the instances of the particular Cartridge cluster.

Each main cartridge Cluster agent will show its statistics data.Also,there will be another graph similar to this for the instance of their Cluster agent. It will show it’s own statistics.





Sketch [2]

This timeline will allow to see the consumption for those instances and clusters and there will be a option to view the log of the equivalent time.

Implementation Details

This section will describe the major components as well as the design decisions and implementations which are a part of this project.The implementation will feature a top-down approach.

Step 1 : Write data to the database
Step 2 : Retrieve data from the database.
Step 3 : Visualising statistics
Step 4 : Visualising Logs

The Database

- Writing Statistics to the Database

The data published by the Message Broker can be written to the RDBMS by a data source which can be created in the Message Broker itself (How to create data source [1]).The database will be created inside the stratos.

- Database

Statistic table,

This table contains all the statistics data of the instances
ex: Instance_Id,Cluster_Id,Load_Average_Event,Memory_Consumption_Event,etc.
*Note: the same data types(data) are used which are published to BAM from the Message Broker using the thrift protocol.

Statistics for the Cluster instances can be read through this table directly and Statistics for the Product Clusters can be retrieved by it's Cluster instance which has the Cluster_Id as a key in the table.

Log_table,

This table includes all the log details for the instances and for the product clusters.
This Includes data such as,Time,Log type(error,info,debug),message etc.

❏ REST API

This API is to expose the health statistics data from the database which is created inside the stratos .It will also enable developers to query the exposed data as well. These features will be implemented in the API.

- Get data from the database for articular Cartridge.
- Get data from the database for articular Cartridge instance.
- Select data from the database for particular Cartridge/Instance.
- Search data in the database for a given Cartridge.

❏ Retrieving data

To retrieve data we can use php and javascriptjson objects are used to feed data to the D3JS, which will make much easy to understand data for the D3JS.

ex: `[{"2008":96},{"2009":74},{"2010":73},{"2011":96},{"2012":124},{"2013":104}]`

❏ Visualising Statistics

Asshown in the sketches there will be 2 different charts for Product clusters and for the cluster instances individually [Sketch 1].Also there will be a timeline which shows Product instance consumption when you zoom out and which shows Cluster instances consumption when you zoom in [Sketch 2].

Due to the collection of data being on the most basic level, it should be possible to support a large amount of customization to the queries that can be run at the user's request.
*Note that this component has higher priority overall compared to the above components. However, the project will attempt it's best to make the visualization much clear and user friendly.

Timeline

The timeline has been devised to implement components as soon as possible, and then to iteratively improve them to the desired level. This ensures that the deliverables are guaranteed, and the iterations allow easier management of the work required on each individual component.

Week	Major Task	Subtasks
May 11- May 17	Stratos Local	● Making the PaaS environment locally. ● Configuring the MB data source. ● Creating the Database
May 18 - May 24	DB retrieving	● Implementing the RDBMS data retrieval using Jaggery,php and javascript.
May 25 - May 31	Creating the REST api	● Implementing querying functions.
June 1 - June 7	Creating the REST api	● Implementing querying functions and exposing data in a accurate way.
June 8 - June 14	Statistic Table Product Cluster	● Making the User Interfaces using D3JS and configuring the JSON
June 15 - June 21	Statistic Table Cluster Instances	● Making the User Interfaces using D3JS and configuring the JSON data.
June 22 - June 28	Time Line for Cluster	● Creating the Ui using D3JS for the timeline using JSON data.
June 29 -July 5	Time Line for Cluster	● Creating the Ui using D3JSS for the timeline using JSON data.
July 6 - July 12	Making log visualization using UI	● Creating the GUI to show data of the logs and

		configuring data.
July 13 - July 19	Making log visualization using UI	• Creating the GUI to show data of the logs and configuring data.
July 20 - July 26	Testing and Bug fixing	• Testing and fixing all the bugs in the code.
July 27 - August 3	Scrub all code and complete testing/documentation	• Add a few more fields to check how extensible system is • Improve extensibility further

About Me

Hello everyone! I'm Ashan Dhananjaya, a 23 year old undergraduate from Informatics Institute of Technology, Sri Lanka. This is my first GSoC experience and I Intend to take part in it next year as well You can contact me through the following detail. Please feel free to leave me any feedback regarding this proposal. cheers!

Email

dhananjaya92@gmail.com

Github

https://github.com/ashandk

Experience

Worked with java,javascript,php,jaggery,jquery,mysql,html etc in the industrial placement.

Contest Winnings

- Imagine Cup 2014 - Runners Up (Gaming Category).

References

[1]=https://docs.wso2.com/display/MB220/Configuring+an+RDBMS+Datasource